Профессия DevOps-инженер PRO

Александр Крупин
Руководитель DevOps-инженеров
в крупной фармацевтической компании

Евгений Дмитриев
DevOps-инженер в ОАО «ИнфоТеКС»

Skillbox

YAML, утилиты, Git

Александр Крупин
Руководитель DevOps-инженеров
в крупной фармацевтической компании

Евгений Дмитриев
DevOps-инженер в ОАО «ИнфоТеКС»

Skillbox

Цели модуля

- Познакомитесь с особенностями YAML-языка
- Узнаете про балансировщики нагрузки, реверс прокси, которые являются одними из основных инструментов в построении больших и отказоустойчивых систем
- Дополним некоторые знания в области Git и рассмотрим ещё один способ построения Gitflow

Синтаксис и особенности YAML

Цели урока

- Понять, как составлять описание конфигураций в формате YAML
- Познакомиться со всеми структурами, списками, особенностями языка
- Узнать, как оптимизировать свои файлы конфигураций и манифестов

Почему YAML?

Файлы конфигурации к некоторому софту, а также различные манифесты описаны на языке **YAML**.

Пример:

- Elasticsearch
- Prometheus
- Docker-compose
- Kubernetes
- GitLab-ci
- Consul
- Sentry и др.

В **Ansible** описывается всё на **YAML**-языке, только **hosts** можно оставить в формате **Toml**.

Типы данных YAML

```
!!null ''                    # null
!!bool 'yes'                 # bool
!!int '3...'                 # number
!!float '3.14...'            # number
!!binary '...base64...'      # buffer
!!timestamp 'YYYY-...'  # date
!!omap [ ... ]               # array of key-value pairs
!!pairs [ ... ]              # array or array pairs
!!set { ... }                # array of objects with given keys and null values
!!str '...'                  # string
!!seq [ ... ]                # array
!!map { ... }                # object
```

Как указывать тип данных?

- foo: !!float 3.231
- str: !!str 42
- tuple_example: !!python/tuple

  - 1234

  - 12
- set_example: !!set {1234, 14}
- date_example: !!timestamp 2021-08-25

Типы данных Python

```
!!python/none          # none
!!python/bool          # bool
!!python/int           # int
!!python/float         # float
!!python/bytes         # bytes
!!python/str           # string
!!python/unicode       # str
!!python/long          # int
!!python/complex       # complex
!!python/list          # list
!!python/tuple         # tuple
!!python/dict          # dict
```

Пример:

```
python_tuple: !!python/tuple
  - foo
  - bar
  - zero
```

Списки (list)

Вариант 1

Записать каждый пункт с новой строки, перед значением необходимо поставить дефис.

Пример:

```
list_one:
  - foo
  - bar
```

Вариант 2

Записать в одну строчку, перечислив все элементы в квадратных скобках через запятую.

Пример:

```
list_one_another: [foo, bar]
```

Словари (dict)

Вариант 1

Каждый ключ мы можем записывать с новой строки, предварительно отступив от левого края на два пробела.

Пример:

```
map_one:
  foo: one
  bar: two
```

Вариант 2

Также можно записать в одну строчку, перечислив все элементы в фигурных скобках.

Пример:

```
map_one_another: {foo: one, bar: two}
```

Строки (strings)

Строки можно записывать по-разному. Строку можно заключить в одинарные или двойные кавычки, а также можно оставить без кавычек.

Пример:

- string_one: Hello SkillBox
- string_two: «Hello SkillBox»
- string_three: ‘Hello SkillBox’

Булево значение (bool)

Булевые значения, такие как правда или ложь, true или false, один или ноль можно записать по-разному.

Пример:

- bool_one: yes
- bool_two: no
- bool_three: True
- bool_four: true
- bool_five: false
- bool_string: «yes»
- bool_dig_yes: 1
- bool_dig_no: 0

Текст (text)

Вариант 1

После двоеточия ставим знак «больше»
и со следующей строки начинаем писать текст,
не забывая про разметку YAML (два пробела от края).

Пример:

```
long_text: >
Support for reading
and
writing

YAML
is available for many programming languages
```

Вариант 2

После двоеточия ставим «пайп»
(или вертикальная черта) и также
набираем текст.

Пример:

```
long_text: |
Support for reading and writing

YAML
is available for many programming
languages
```

Переменные (environments)

Мы можем сделать ключ, поместить в него разные списки, текста, другие словари и использовать этот ключ как переменную в разных местах YAML-файла.

Пример:

```
foo: &default_settings
  env: dev
  db:
     host: localhost
db_name: test
port: 5432
```

Запись переменных. Вариант 1

Необходимо написать ключ и после двоеточия поставить знак астериска (* | звёздочку) и слитно написать имя переменной.

Пример:

dev: *default_settings

stage: *default_settings

Запись переменных. Вариант 2

Пишем ключ, ставим двоеточие на следующей строке после отступа, пишем два знака «меньше», ставим двоеточие и также через астериск (* | звёздочка) набираем название переменной.

Пример:

```
stage:
  <<: *default_settings
  env: stage
  app:
    port: 8080
```

Применение переменных

```
.build_image: &build_image
  stage: build
  image:
    name: gcr.io/kaniko-project/executor:debug
  script:
    - /kaniko/executor --destination $build_image

image:k8s:
  <<: *build_image
  variables:
    build_image: registry.com/${CI_COMMIT_SHA}:${CI_PIPELINE_ID}
  rules:
    - if: $CI_COMMIT_BRANCH == 'master'
```

Multiple documents

```
---
foo: «Hello SkillBox»
---
foo: «Hello Students»
```

Итоги урока

- Познакомились со структурой YAML-файла, переменными и типами данных
- Рассмотрели особенности языка